

CLAIM AMENDMENTS

Please amend claim 10 as described herein below.

Other than claim 10, no amendments to the claims have been knowingly made.

1(ORIGINAL). In a device that uses a java programming language and a second programming language, a method of emulating more simultaneously open files in the device than is permitted by an operating system of the device comprising:

- while running a predetermined program in the java programming language, attempting to access a file stored within the device;
- determining whether the file is already open from prior execution;
- if the file is open, access to the file proceeds;

- if the file is not open, a determination is made by the device of a number of files the device permits to be simultaneously open;
- if the number is not exceeded by opening the file, the file is opened;

- *if the number is exceeded by opening the file, the device emulates having more files open than the number allowed by:

- (1) saving a file position for at least one open file, the file position designating where a next byte in the at least one open file would be accessed;

- (2) closing the at least one open file; and

- (3) opening the file;

wherein subsequent accesses to the at least one open file that has been closed are made transparently to the predetermined program in the java programming language by closing at least one currently open file, retrieving the file position previously saved, and opening and restoring the file position for that file.

2(ORIGINAL). The method of claim 1 further comprising:

saving the file position in storage allocated for usage by java code.

3(ORIGINAL). The method of claim 1 further comprising:

implementing the second programming language as c programming language.

4(ORIGINAL). The method of claim 1 further comprising:

providing a table;

storing arbitrarily long file names in a first format;

storing corresponding shortened names in a second format not

exceeding a maximum number of characters; and

translating from the first format to the second format when a file is

initially opened under control of the java programming

language, the translating being implemented transparent to any

java programming language application so that the java

programming language does not utilize the corresponding

shortened names.

5(ORIGINAL). The method of claim 4 further comprising:

performing the translating only when the operating system does not

permit file names having a length exceeding a predetermined

maximum number of characters.

6(ORIGINAL). The method of claim 4 further comprising:

performing the translating only when the operating system does not

support first format file names.

7(ORIGINAL). The method of claim 4 further comprising:
placing a unique identifier within predetermined positions of the
shortened names to identify which file is being retrieved by the
operating system.

8(ORIGINAL). The method of claim 7 further comprising:
using the unique identifier to bypass the translating of the first format
to the second format when a predetermined file is reopened
after being previously closed.

9(ORIGINAL). The method of claim 1 further comprising:
providing a table;
storing a directory in the table, the directory indicating a java
language application suite corresponding to a predetermined
file; and
using the directory in the table to distinguish files of the same name
belonging to different java language application suites.

10(CURRENTLY AMENDED). The method of claim 1 further comprising:
using storage allocated for use by java code to contain variables such
as ~~file position~~ needed by the second programming language.

11(ORIGINAL). The method of claim 10 further comprising:
using the variables to track the current position within an open file and
to reopen and reposition the file if the file is temporarily closed.

12(ORIGINAL). The method of claim 10 further comprising:
using automatic memory management features of the java
programming language including garbage collection to reclaim
storage when the file is no longer in use.

13(ORIGINAL). The method of claim 1 further comprising:
implementing the device as a portable, wireless device.

14(ORIGINAL). A method of interfacing a file system in a device that uses
both a java programming language and another programming language,
comprising:
selecting a predetermined operating system;
overlying a java virtual machine for executing programs within the
operating system in the another programming language;
overlying the operating system with a file system interface layer;
overlying libraries to the java virtual machine and the file system
interface layer, the libraries running in the java programming
language;
using the file system interface to determine if a maximum number of
open java files has been exceeded, and if so, storing position
information of at least one open java file in a currently open
java application prior to closing the at least one open java file;
and
subsequently reopening the at least one open java file by restoring the
position information transparent to any java application.

15(ORIGINAL). The method of claim 14 further comprising:
providing a table;
storing a directory in the table, the directory indicating a java
language application suite corresponding to a predetermined
file; and
using the directory in the table to distinguish files of the same name
belonging to different java language application suites.

16(ORIGINAL). The method of claim 15 further comprising:
using storage allocated for use by java code to store variables needed
by the another programming language.

17(ORIGINAL). The method of claim 16 further comprising:
using the variables to track the current position within an open file and
to reopen and reposition the file if the file is temporarily closed.

18(ORIGINAL). The method of claim 14 further comprising:
providing a table;
storing arbitrarily long file names in a first format;
storing corresponding shortened names in a second format not
exceeding a maximum number of characters; and
translating from the first format to the second format when a file is
initially opened under control of the java programming
language, the translating being implemented transparent to any
java programming language application so that the java
programming language does not utilize the corresponding
shortened names.

19(ORIGINAL). The method of claim 18 further comprising:
placing a unique identifier within predetermined positions of the
shortened names to identify which file is being retrieved by the
operating system.

20(ORIGINAL). The method of claim 19 further comprising:
using the unique identifier to bypass the translating of the first format
to the second format when a predetermined file is reopened
after being previously closed.